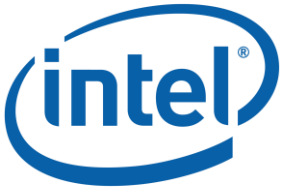


MayLoon User Manual

Copyright © 2013 Intel Corporation

Document Number: xxxxxx-xxxUS

World Wide Web: http://www.intel.com/software

Document Number: XXXXX-XXXXX

Disclaimer and Legal Information

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER, AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT. UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.
Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.
The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.
Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.
Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's Web Site.

Intel processor numbers are not a measure of performance. Processor numbers differentiate features within each processor family, not across different processor families. See http://www.intel.com/products/processor_number for details.

BunnyPeople, Celeron, Celeron Inside, Centrino, Centrino Atom, Centrino Atom Inside, Centrino Inside, Centrino logo, Core Inside, FlashFile, i960, InstantIP, Intel, Intel logo, Intel386, Intel486, IntelDX2, IntelDX4, IntelSX2, Intel Atom, Intel Atom Inside, Intel Core, Intel Inside, Intel Inside logo, Intel. Leap ahead., Intel. Leap ahead. logo, Intel NetBurst, Intel NetMerge, Intel NetStructure, Intel SingleDriver, Intel SpeedStep, Intel StrataFlash, Intel Viiv, Intel vPro, Intel XScale, Itanium, Itanium Inside, MCS, MMX, Oplus, OverDrive, PDCharm, Pentium, Pentium Inside, skoool, Sound Mark, The Journey Inside, Viiv Inside, vPro Inside, VTune, Xeon, and Xeon Inside are trademarks of Intel Corporation in the U.S. and other countries.

* Other names and brands may be claimed as the property of others.

Copyright (C) 2010, Intel Corporation. All rights reserved.

Revision History

Document Number	Revision Number	Description	Revision Date
xxxxx	001	Initial release.	March 2013
xxxxx	002	Updated for MayLoon alpha	May 2013
	003	Updated for MayLoon 0.3.1	June 2013
	004	Updated for MayLoon 0.4	July 2013

Table of Contents

1 Environment Requirements 4

2 Installing MayLoon Porting Tool 5

3 Uninstalling and Upgrading Mayloon Porting Tool 8

3.1 Uninstall 8

3.2 Upgrade 9

4 Configuring MayLoon Porting Tool 10

5 Using MayLoon Porting Tool 11

5.1 Converting to MayLoon 11

5.2 Implementing Stubs 15

5.3 Building and Running 17

5.4 Debugging Application in Browser 19

5.5 Packaging for Tizen 22

6 Supported APIs 28

7 Unsupported Features 29

8 Known Incompatibilities and Limitations 31

8.1 Eclipse plugin 31

8.2 Incompatibility between Java and JavaScript 31

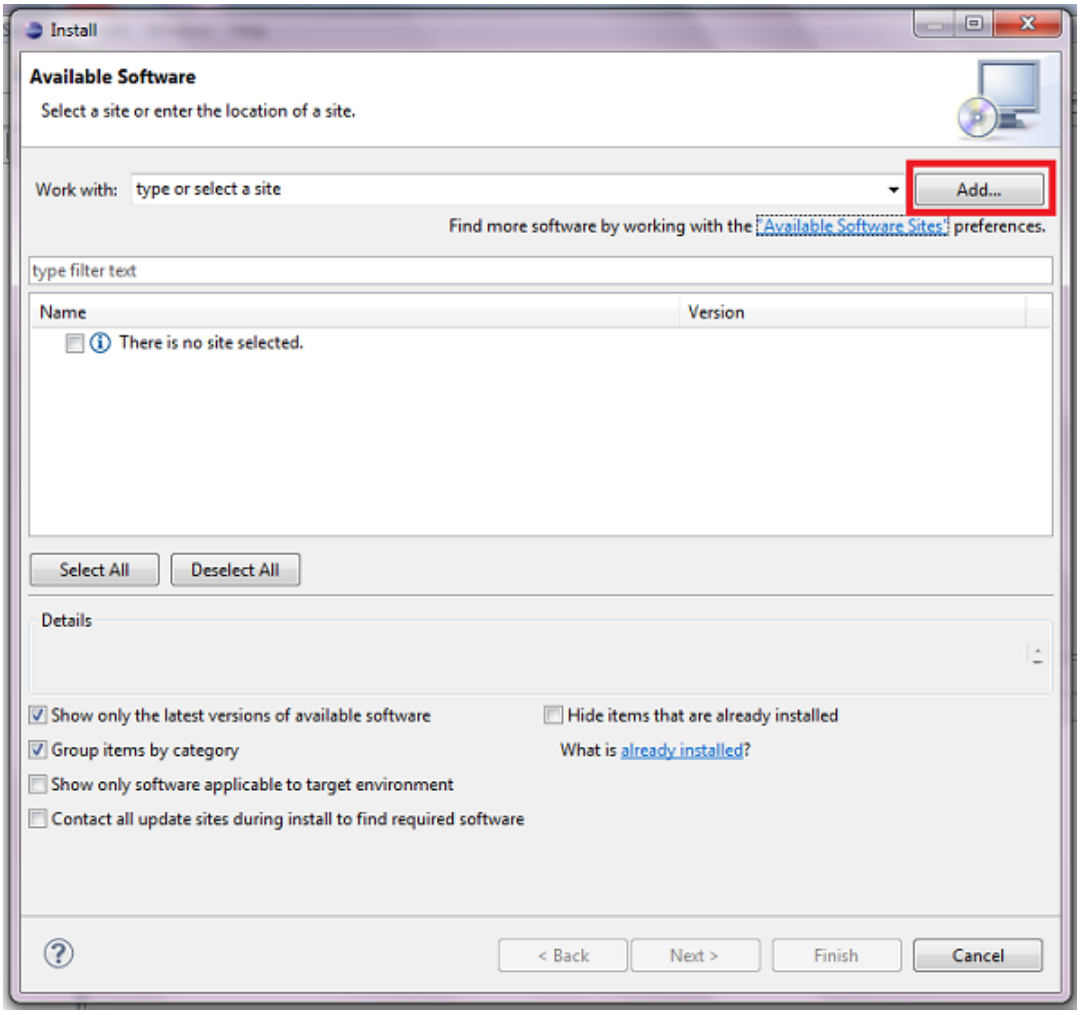
9 Upcoming Features 33

1 Environment Requirements

- Linux or Windows PC
- Java JDK 1.6
- Eclipse Classical 3.7.2 (Indigo SR2) or higher with android ADT 21 or greater and JDT plugin installed
- MayLoon porting tool installation package (MPT.zip)
- MayLoon SDK archive file (MayLoon.zip)

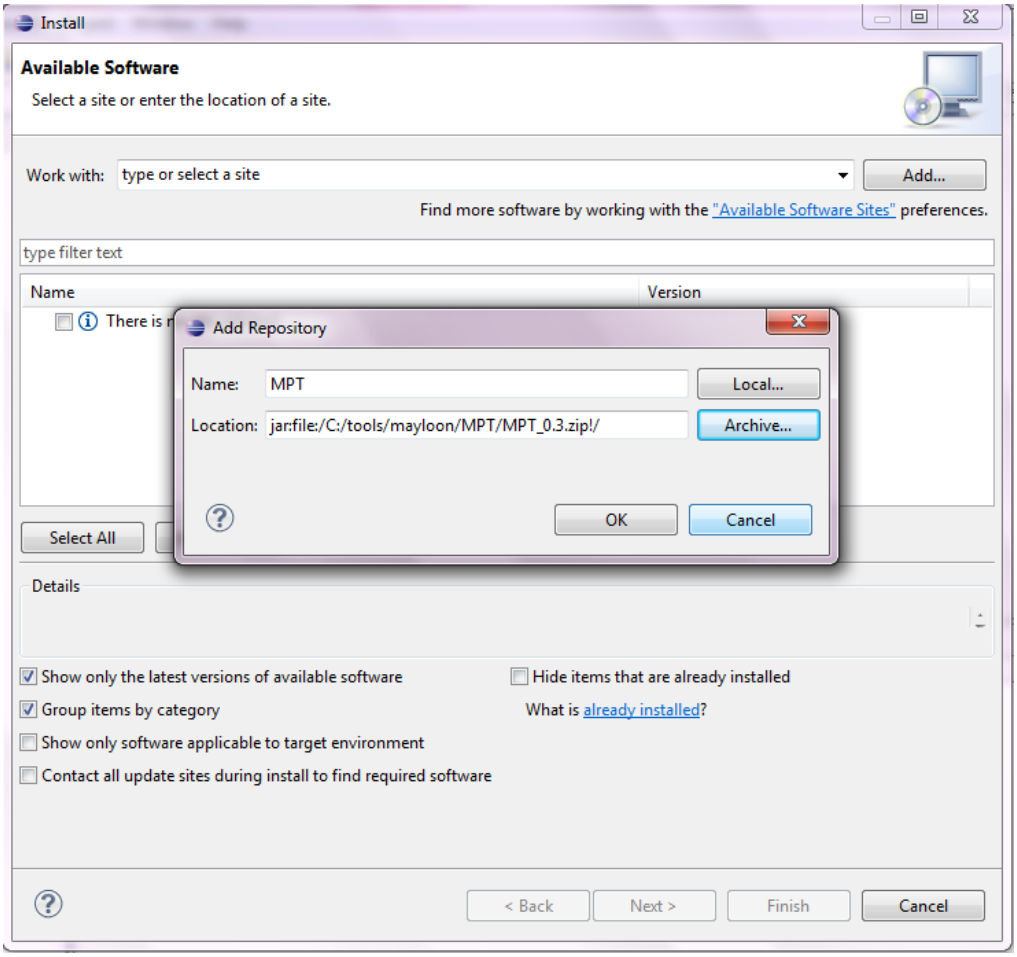
2 Installing MayLoon Porting Tool

1. Open Eclipse IDE, Help-> Install New Software..., then click "Add..." button


Install
Available Software
Select a site or enter the location of a site.
Work with:
type or select a site
Add...
Find more software by working with the Available Software Sites preferences.
type filter text
Name
Version
There is no site selected.
Select All
Deselect All
Details
Show only the latest versions of available software
Group items by category
Show only software applicable to target environment
Contact all update sites during install to find required software
Hide items that are already installed
What is already installed?
< Back
Next >
Finish
Cancel

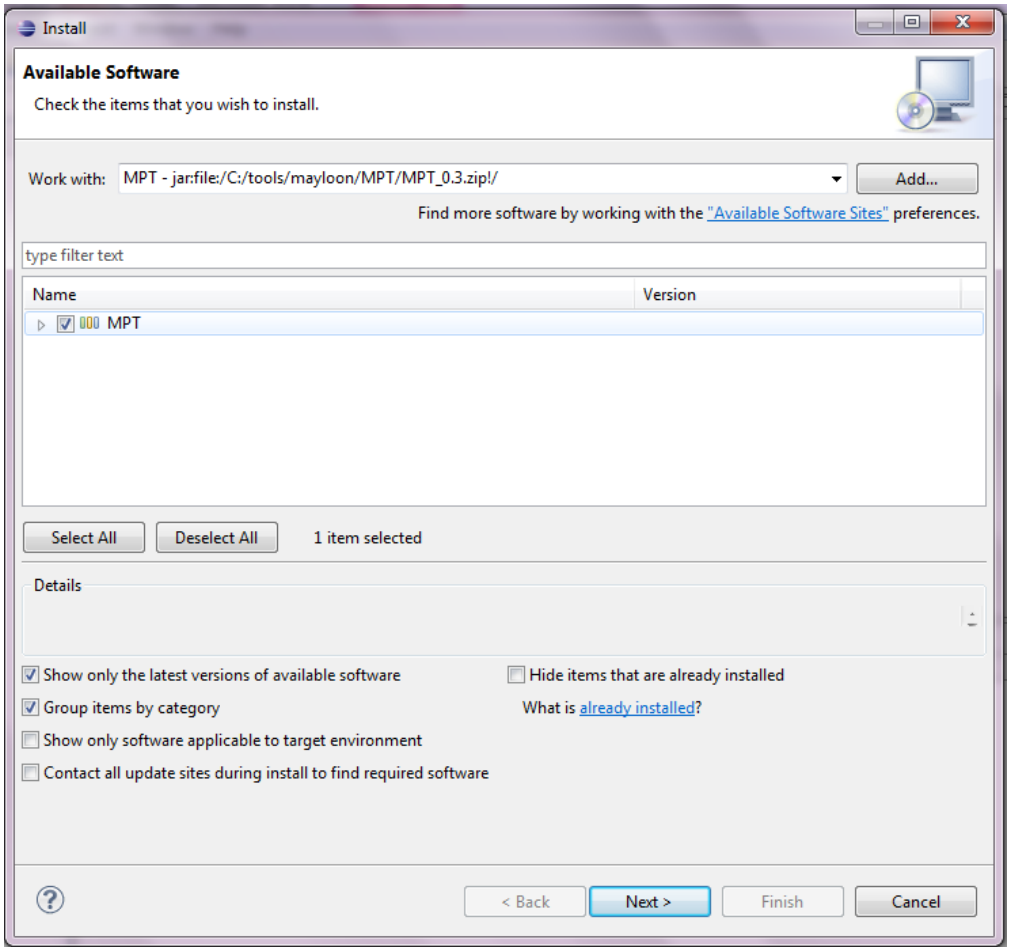
Pic1 Install MPT

2. Click Archive..., then select MPT_xxx.zip and OK



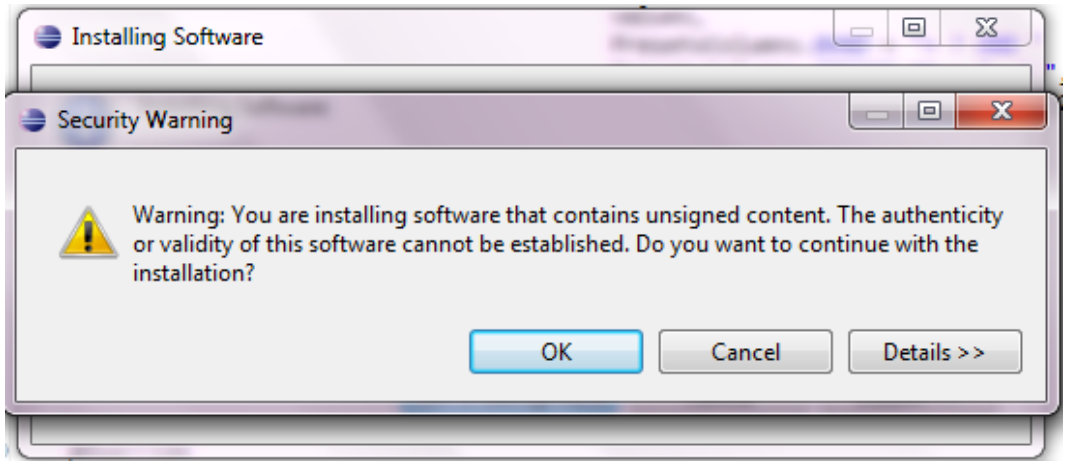
Pic2 Add installation package

3. Install MPT feature as normal eclipse plugin.



Pic3 Installation process

Note: If you get a security warning saying that the authenticity or validity of the software can't be established, click OK and continue.

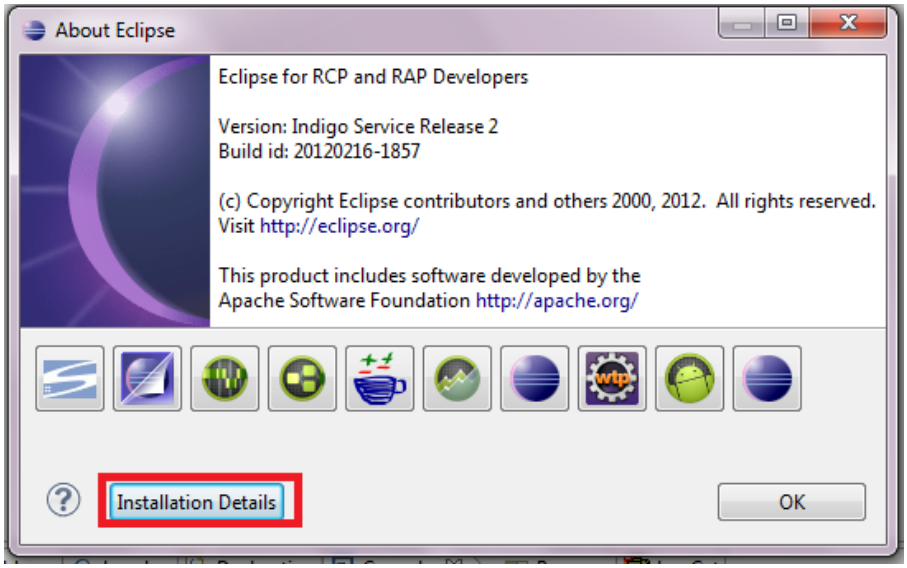

Installing Software
Security Warning
Warning: You are installing software that contains unsigned content. The authenticity or validity of this software cannot be established. Do you want to continue with the installation?
OK
Cancel
Details >>

Pic4 Security warning

3 Uninstalling and Upgrading Mayloon Porting Tool

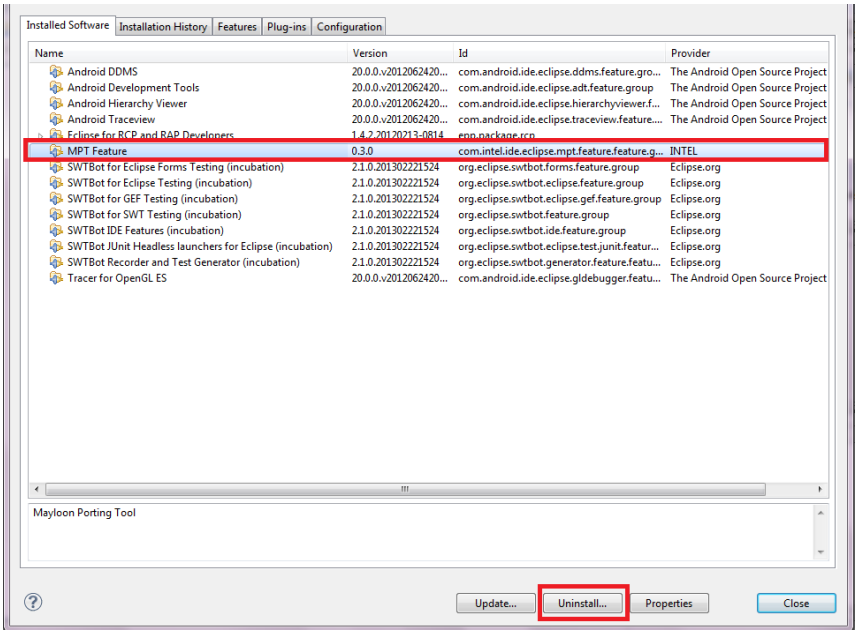
3.1 Uninstall

1. Click Help->About Eclipse, then click Installation Details button in popup window:


About Eclipse
Eclipse for RCP and RAP Developers
Version: Indigo Service Release 2
Build id: 20120216-1857
(c) Copyright Eclipse contributors and others 2000, 2012. All rights reserved.
Visit http://eclipse.org/
This product includes software developed by the
Apache Software Foundation http://apache.org/
Installation Details
OK

Pic5 Uninstall 1

2. Select MayLoon porting tool plugin in Installed Software Tab and then click Uninstall… button:



Pic6 Uninstall 2

3. Then click Finish button to finish uninstalling and click Restart Now button to make uninstallation work immediately

3.2 Upgrade

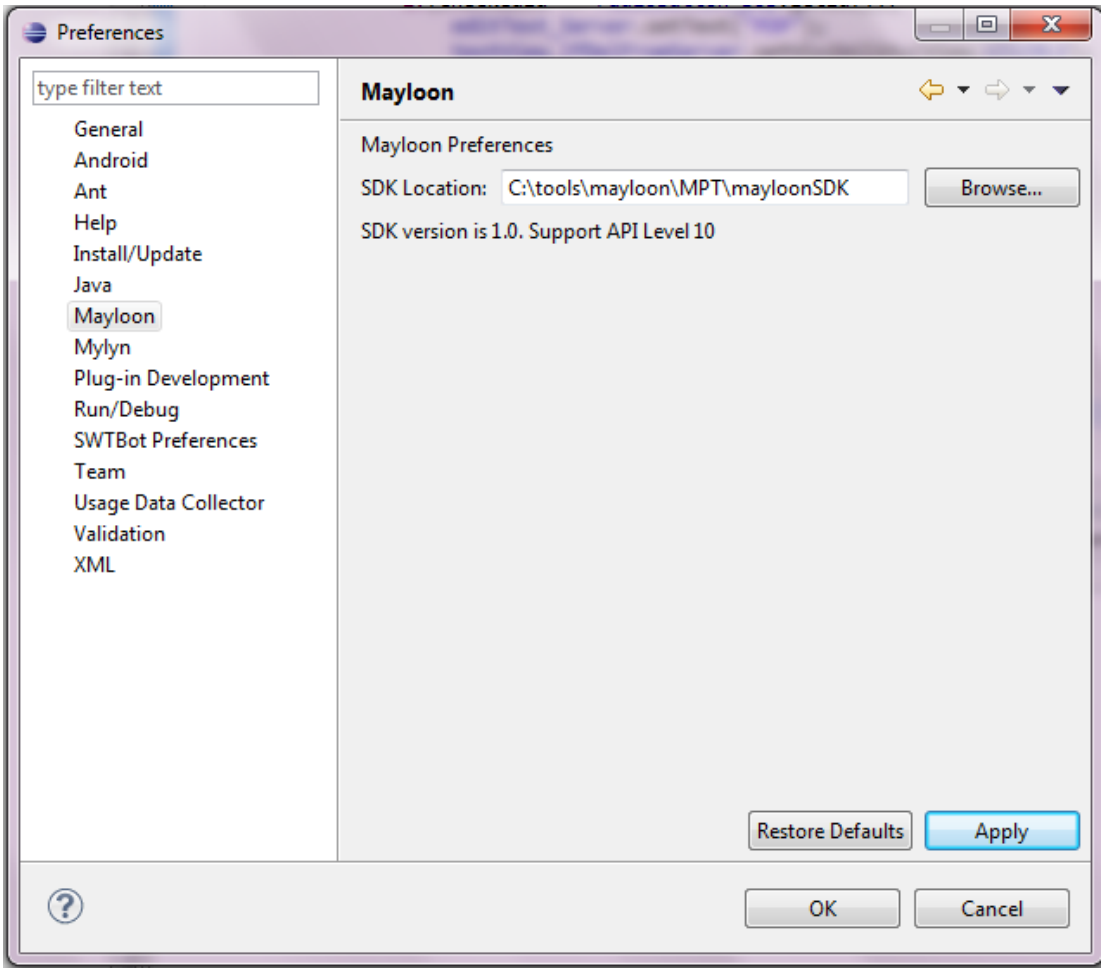
If you want to Upgrade MayLoon porting tool, please uninstall installed MayLoon porting tool plugin as 3.1 describes, and then follow section 2 to install your new MayLoon porting tool.

4 Configuring MayLoon Porting Tool

Click Window->Preferences, select Mayloon on the left tab, there will be 1 item to configure:

SDK Location: Set proper location of MayLoon SDK (folder where MayLoon.zip is extracted) as the following pic shows

Click Apply and OK button to apply your settings.



Pic7 Mayloon preferences

5 Using MayLoon Porting Tool

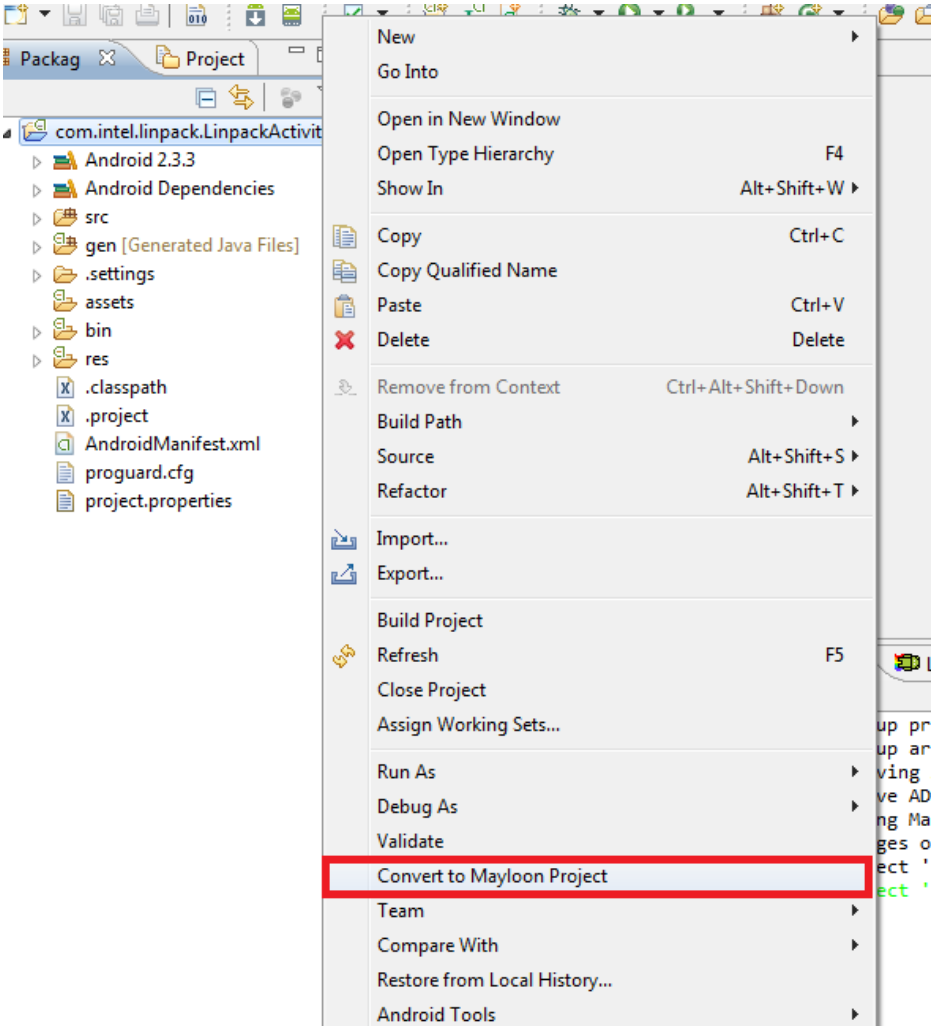
This section goes through the steps to convert and build your Android application to HTML5.

5.1 Converting to MayLoon

1. Import an existing Android project into Eclipse

2. Build the Android project; make sure an android application package is generated successfully, usually under the bin/ directory of the Android project. **Note** if the application uses proprietary extensions to Android framework and cannot build in Eclipse environment, please generate an APK package corresponding to this application offline, and put it in the bin/ directory. The conversion cannot proceed without the APK file.

3. In Eclipse IDE, Select your project to convert, and then right click. Click "Convert to Mayloon Project".



Pic8 Convert to Mayloon project

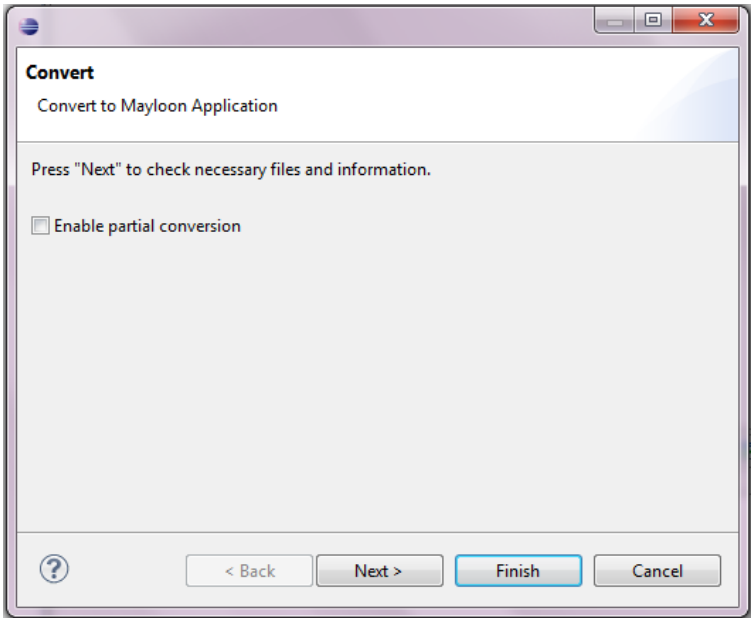
4. Convert Wizard will pop up, you can configure partial conversion mode for application.

Partial Conversion Mode: If partial conversion is enabled, MayLoon will modify application source code in the following way during conversion:

- It converts JNI method declarations into stub methods that you can later implement in either Java or JavaScript.
- Under this mode, MayLoon will also generate stub classes for the Android framework classes that MayLoon does not support.

Select the checkbox to enable this mode.

Click Next> button to continue.

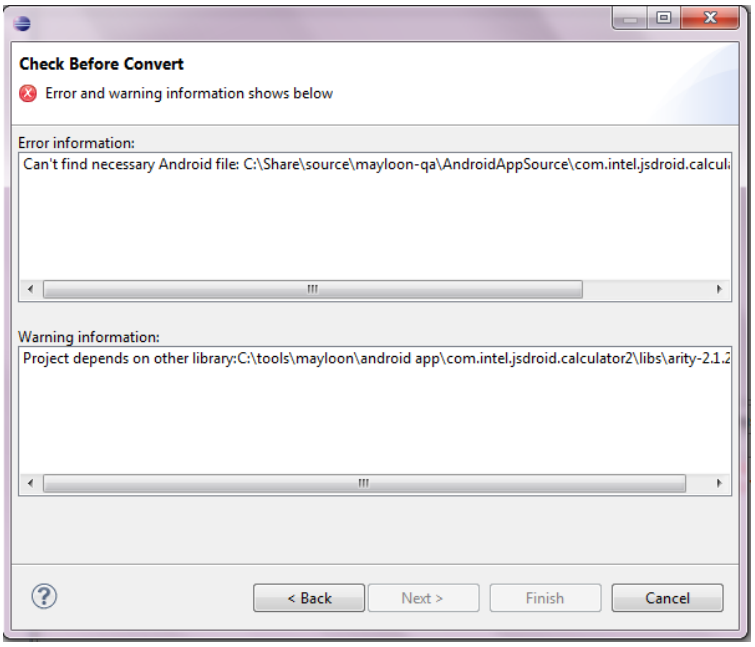


Pic9 Convert wizard

5. Check info will list in this page.

   Error information: Errors that will terminate convert process

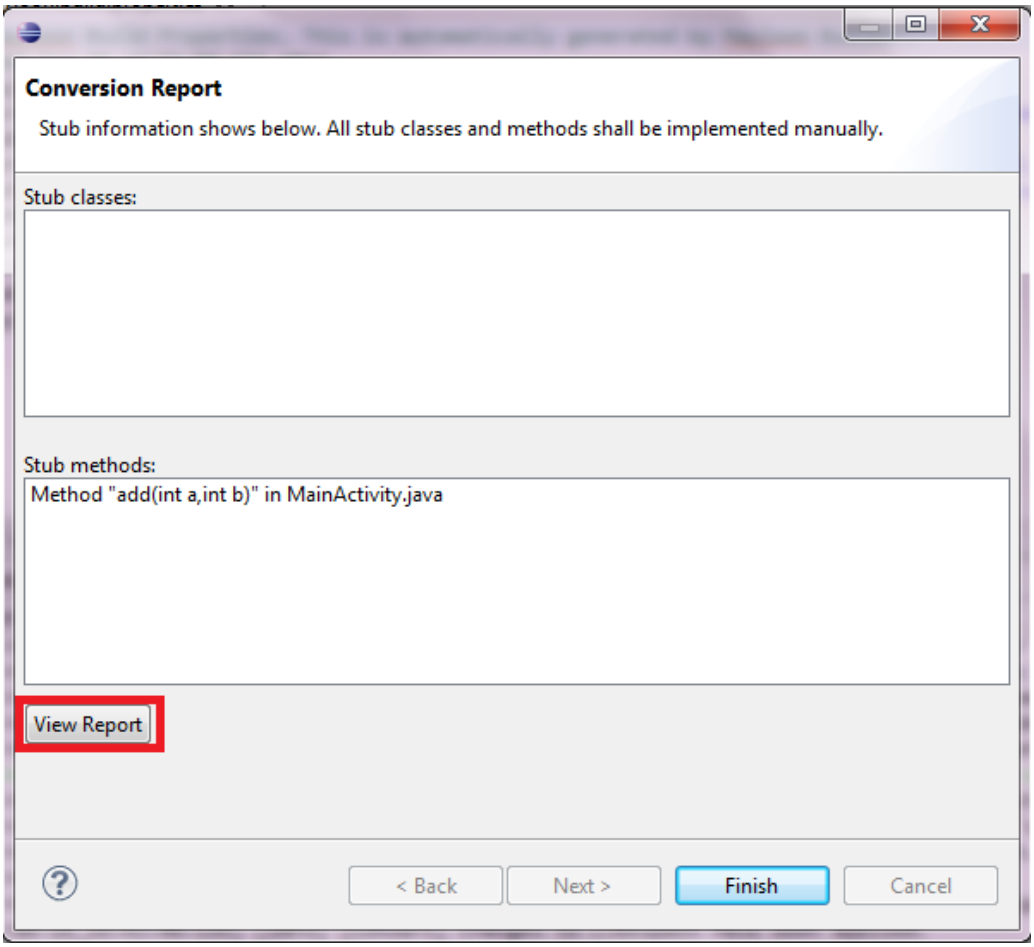
   Warning information: Information that needs developer to pay attention to



Pic10 Check before conversion

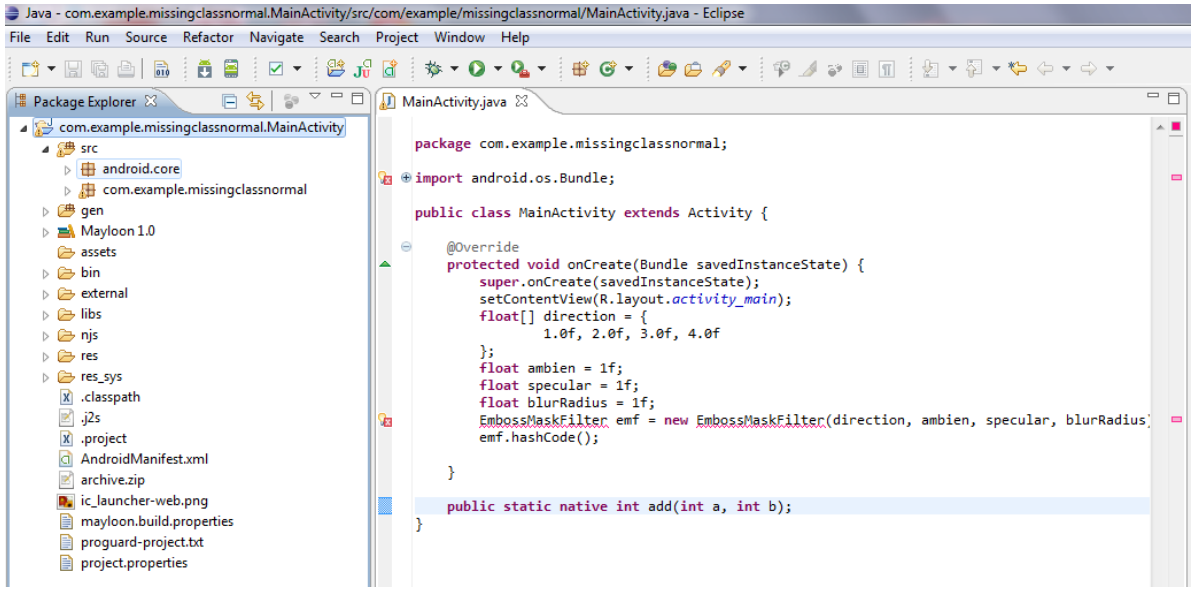
Click Next> or Finish button if there is no error in project, otherwise please fix the error before converting project.

6. If partial conversion mode is enabled, stub class and method info will be listed. You can also click View Report button to view stub info in browser.



Pic11 Conversion report

7. **Note** that after conversion, the project is still a Java project. When you run it as MayLoon project, it will be translated into an HTML5 application. After converting Android application to MayLoon project, if Partial Conversion mode is disabled, and converted application contains APIs/classes not supported by Mayloon, problem markers will be displayed in editor as the following pic shows:

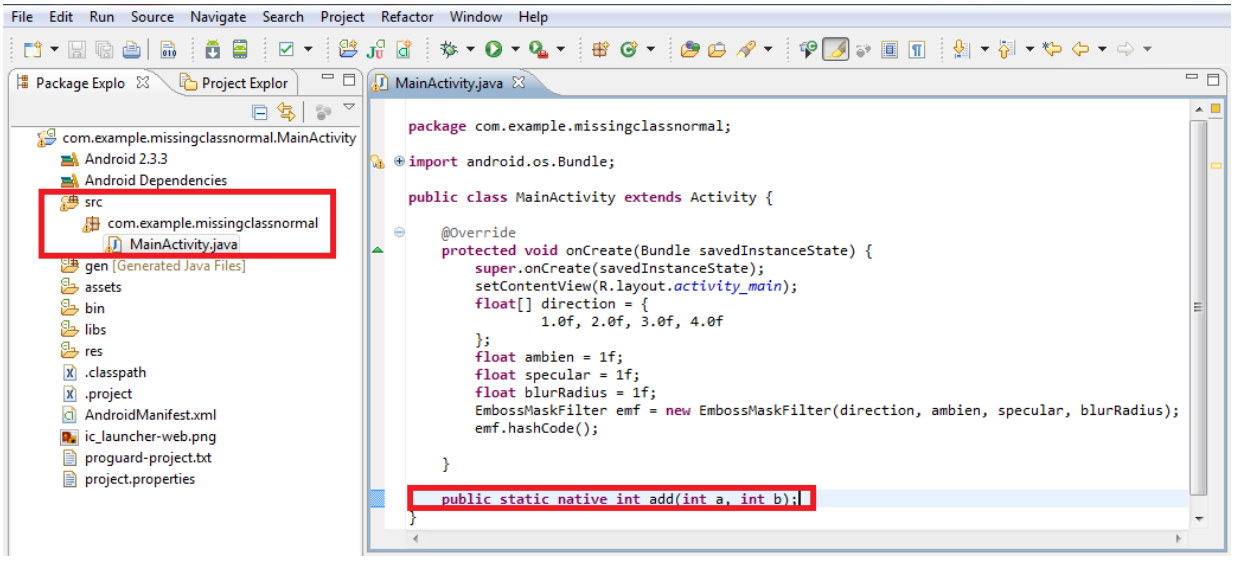


Pic12 Converted MayLoon project without partial conversion

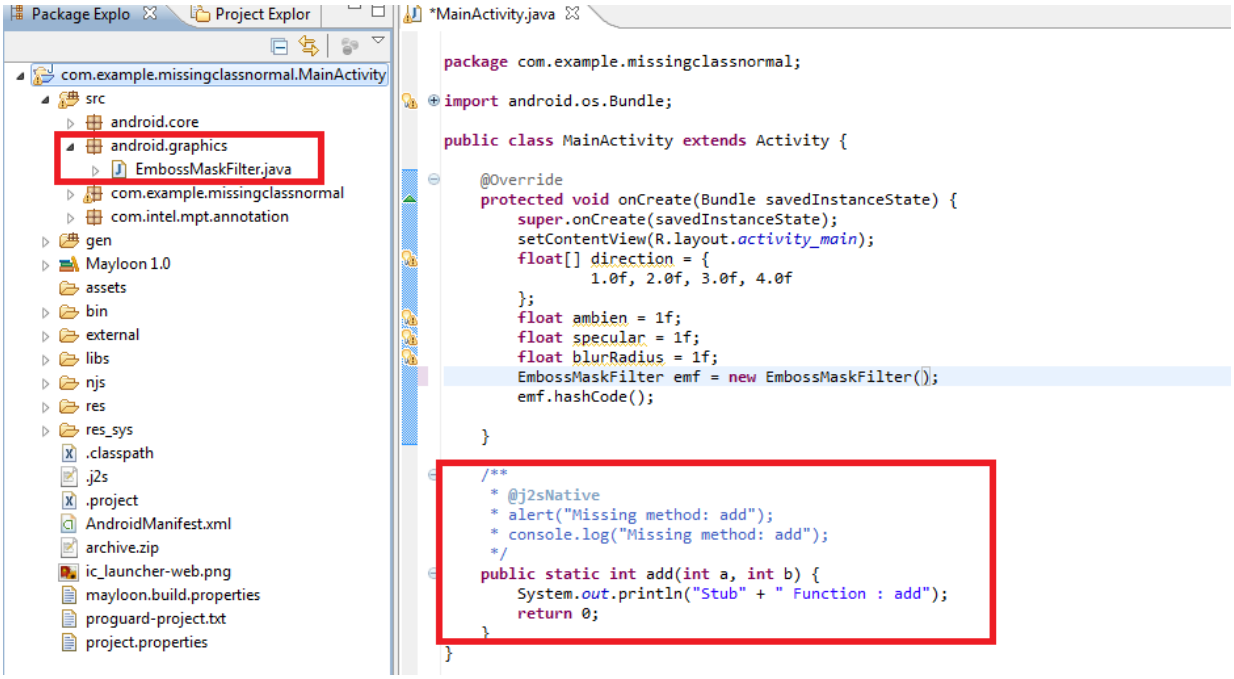
If Partial Conversion mode is enabled, stub classes/functions will be generated for classes not supported by Mayloon and JNI method declarations. Refer to section 5.2 for details. You can review the generated code before proceeding.

5.2 Implementing Stubs

If Partial Conversion mode is enabled, stub classes/functions will be generated for classes not supported by Mayloon and JNI method declarations. Here is a comparison between original android project and converted MayLoon project with partial conversion:



Pic13 Original Android project



Pic14 Converted MayLoon project with partial conversion enabled

A list of the generated stub methods and stub classes are printed in the Eclipse console. You can review the generated stubs and implement them:

- To implement the method add(int a, int b), please remove the comment including the @j2sNative just above the method, and implement the method using either Java code or JavaScript. Intuitively, the above method would be implemented as follows:

```
public static int add(int a, int b){
    return a + b;
}
```

- The generated stub class EmbossMaskFilter contains 3 constructors that once called will error out. To implement the stub class, please implement all the necessary constructors and methods as in normal Android / Java application. If the stub class originates from Android framework, please refer to the official Android developer documentation - http://developer.android.com/index.html for detailed description of the class.

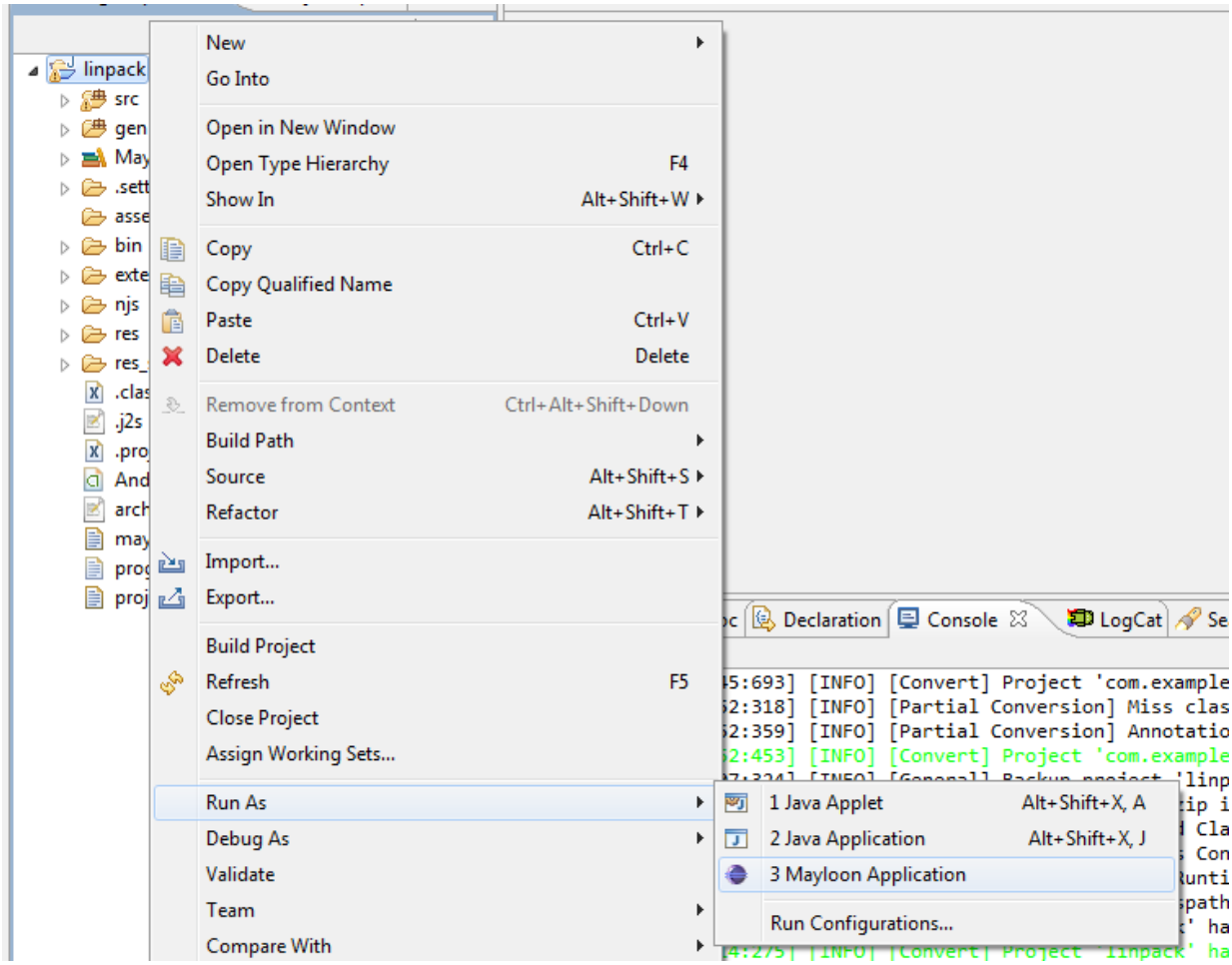
In addition to converting Java code to JavaScript, MayLoon also offers a hybrid programming model, meaning that you can embed JavaScript code into Java program, in the same way as you can embed assembly code into C/C++ program. MayLoon uses the Java2Script library to achieve hybrid programming. The @j2sNative block in Pic14 is an example illustrating how this works. Please refer to http://j2s.sourceforge.net/ for more information on hybrid programming using j2s.

Application developers can use hybrid programming for implementing stub classes and methods.

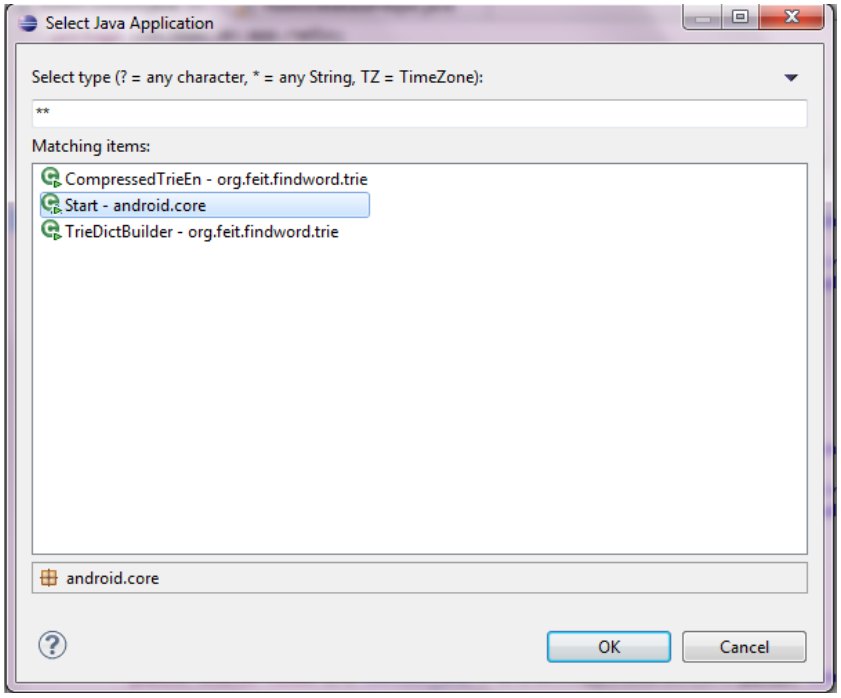
After the stubs are implemented as necessary, the project is ready to be translated into HTML5.

5.3 Building and Running

1. Finally, you can click "Run As" from the popup menu, and then click "Mayloon Application". If there are one or more main() entries in your application, **Select Java Application** dialog will pop up, select item **Start-android.core** and click OK to continue. In this step, all the Java source code files will be translated into JavaScript code, and an HTML file will be generated as the entry of converted web application.

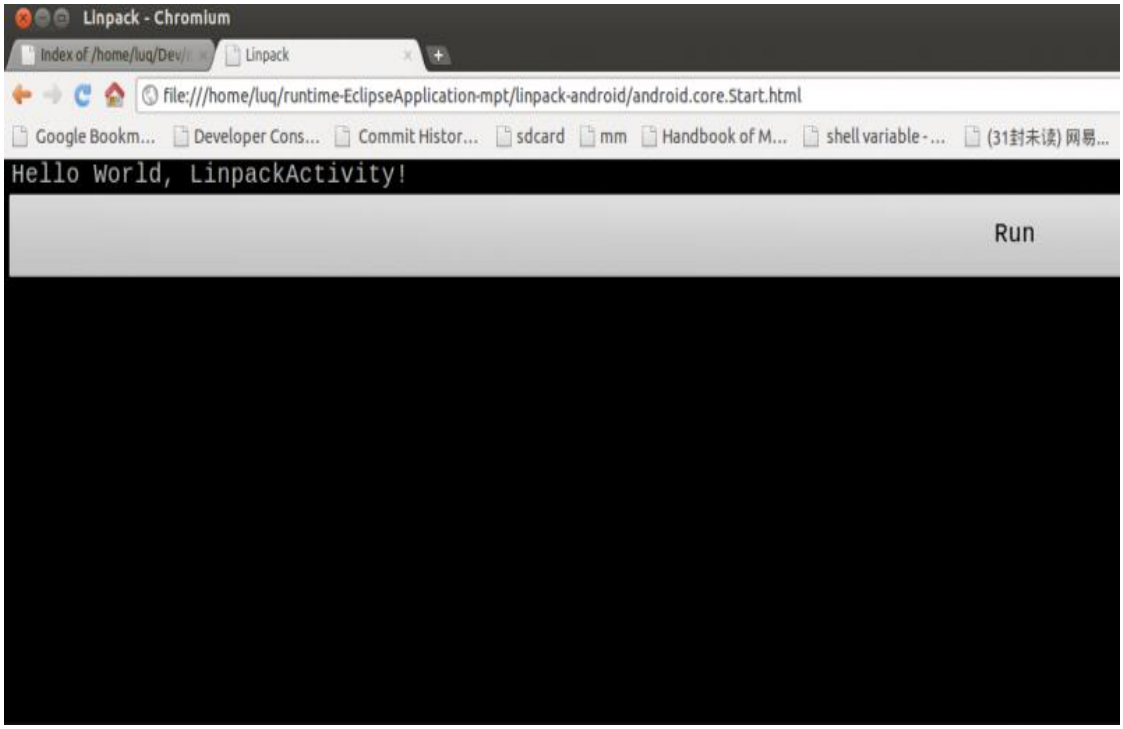


Pic15 Run as Mayloon application



Pic16 Select Java application to run as MayLoon application

2. Start your desktop Chrome browser with the "--allow-file-access-from-files" command line option, and open the generated start page. If a Chrome instance is already running without this option, please close that first. Please check the log messages in the Developer Console of Chrome in case the application does not launch successfully.

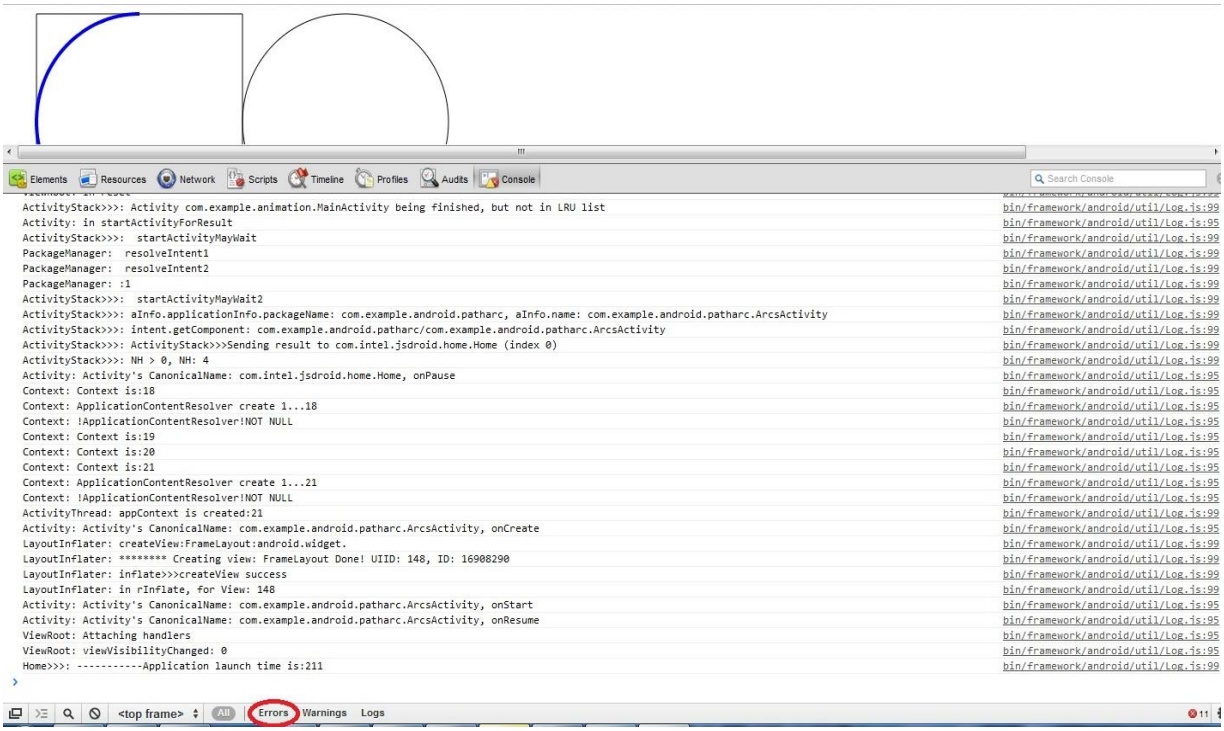


Pic17 Converted application running on Chrome

5.4 Debugging Application in Browser

If you fail to launch your converted application in browser, you can follow the below steps to identify the problem:

1. Take Chrome / Chromium browser as an example, press F12 to invoke the "Developer Tools" view, and click "Console" tab as below to check the console message, you can filter it with "Errors" at the bottom of the "Console" tab:



Pic18 Chrome console

1. As the errors in "Console" are printed out when the errors already occur, if you want to pause at the stack when errors occur you can switch to "Sources"(It can be "Scripts" varying from Chromium versions) tab and enable "Pause all exceptions" at the bottom. Then press F5 to refresh the page to reload the application, and it will pause at the exceptions.

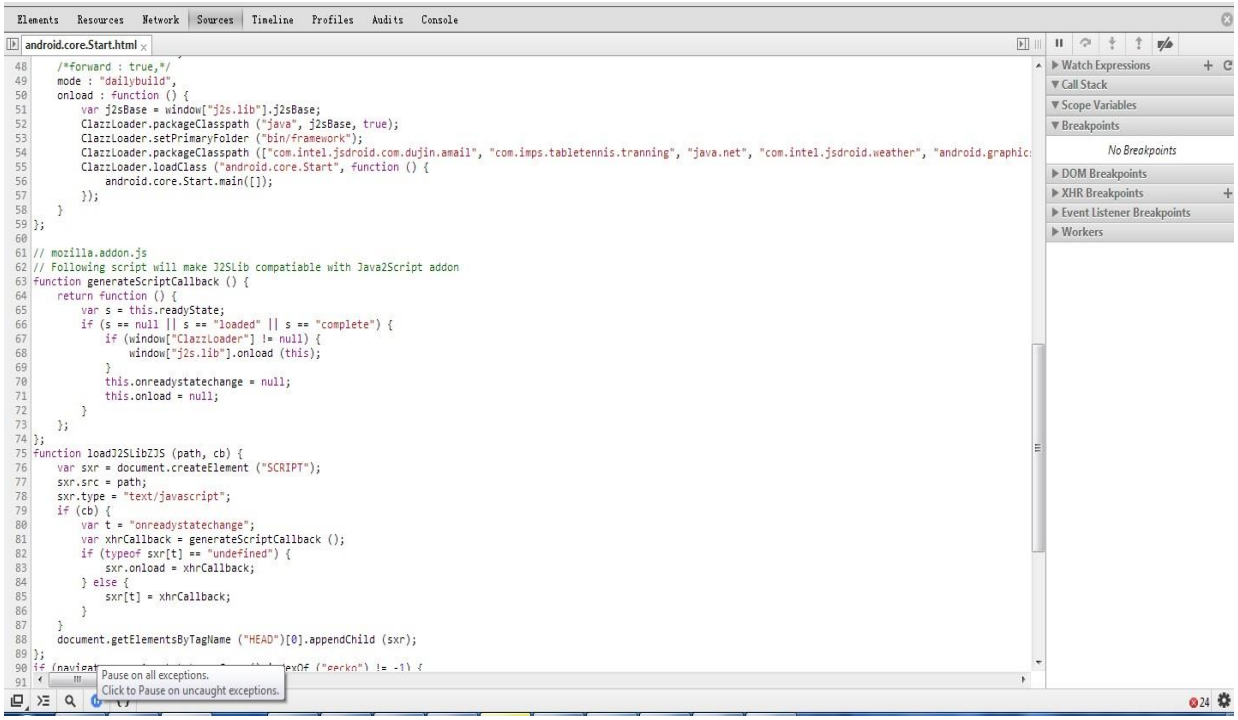
   Other than relying on runtime exceptions, MayLoon also provides the force breakpoints for the developer. It will force the debugger to break at that line if you have opened the Chrome Inspector, regardless there is exception or not. You can insert the breakpoint by using either of the below functions in Java code:

   DebugUtils.DebugBreakPoint(); // for unconditional breakpoint

   DebugUtils.DebugBreakPoint(boolean condition); // for conditional breakpoint

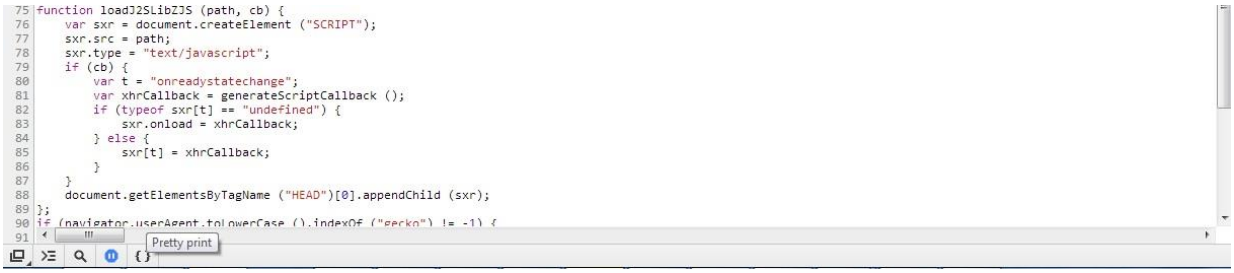
   If you want to force break at JavaScript level, you can use below code:

   debugger; // break at JavaScript level



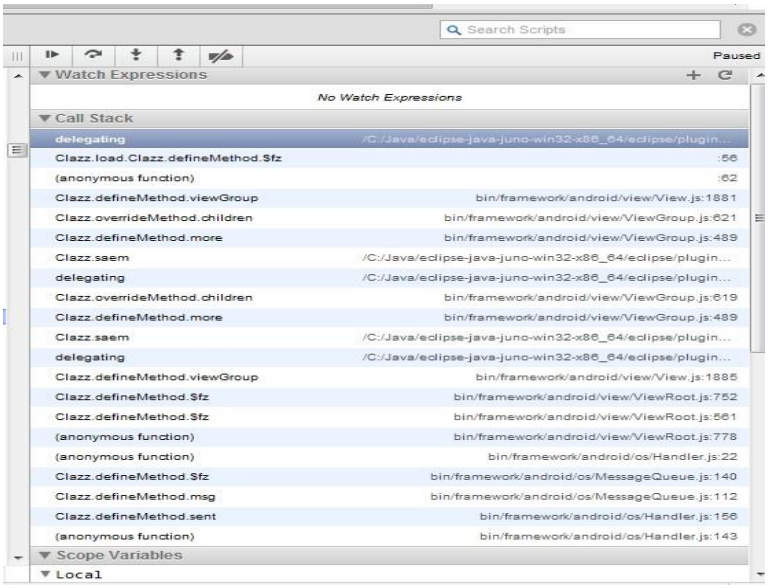
Pic19 Chrome console debugging

2. You can click "Pretty Print" to make the JavaScript code in the panel more readable (with proper indent).



Pic20 Chrome console format

3. You can check "Call Stack", "Watch Expressions", "Scope Variables" and also "Step into the function" in the Debug Area at the right of the source code panel. You can also search methods in the "Search Scripts" toolbar.



Pic21 Stack info

4. Sometimes we can't step into the function directly. We need to reach the function by going through some internal low-level function calls (This is depending on the compiler's implementation).

```
419     Clazz.saem = function(To, claxxRef, fxName, funParams) {
420         var params = Clazz.getParamsType(funParams);
421         var fx = To[fxName];
422         if (fx.lastParams == params.typeString && fx.lastClaxxRef === claxxRef) {
423             var methodParams = null;
424             if (params.hasCastedNull) {
425                 methodParams = new Array();
426                 for (var k = 0; k < funParams.length; k++) {
427                     if (funParams[k] instanceof Clazz.CastedNull) {
428                         methodParams[k] = null;
429                     } else {
430                         methodParams[k] = funParams[k];
431                     }
432                 }
433             } else {
434                 methodParams = funParams;
435             }
436             if (fx.lastMethod != null) {
                 return fx.lastMethod.apply(To, methodParams);
438             } else {
439                 return;
440             }
441         }
442         fx.lastParams = params.typeString;
443         fx.lastClaxxRef = claxxRef;
444         var stacks = fx.stacks;
445         if (stacks == null) {
```

Pic22 Application debug

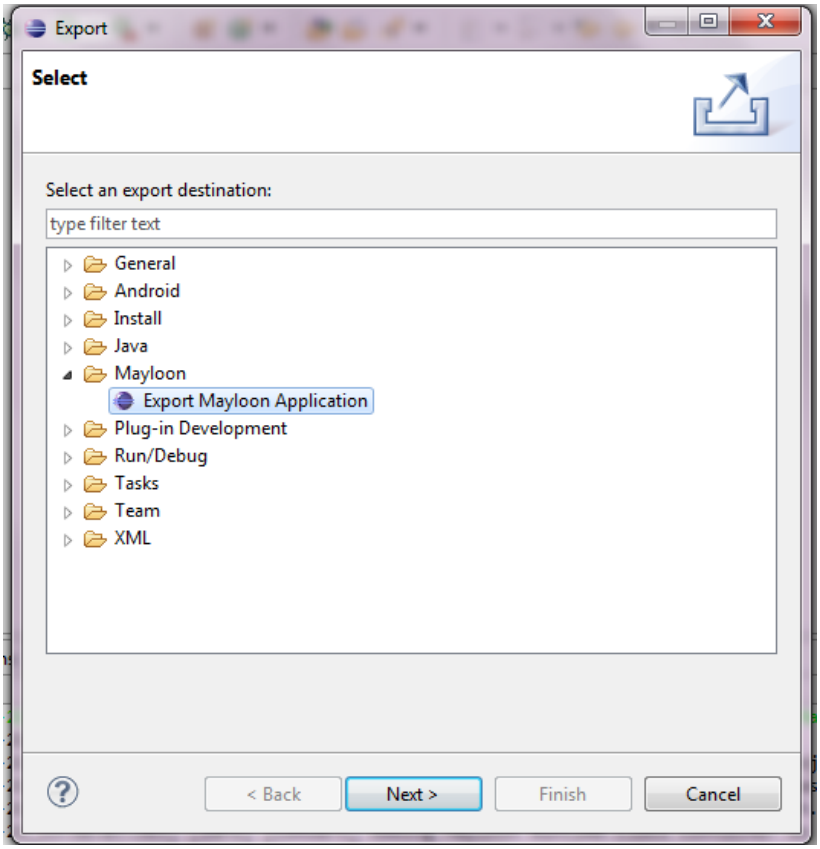
5.5 Packaging for Tizen

If you want to package the converted application for Tizen, please follow the steps below after converting the project to MayLoon:

1. Open mayloon.build.properties file of your converted application, modify value of mayloon.deploy.mode to "tizen":

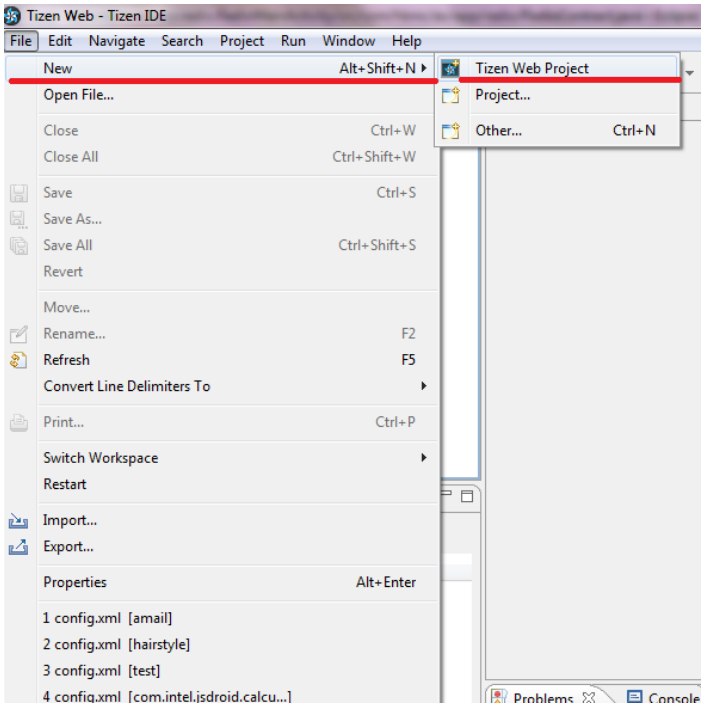
   mayloon.deploy.mode=tizen

2. Right click converted project and select Run As->Mayloon Application
3. Right click project and then select Export...
4. Expand Mayloon and select Export Mayloon Application and click Next as follows



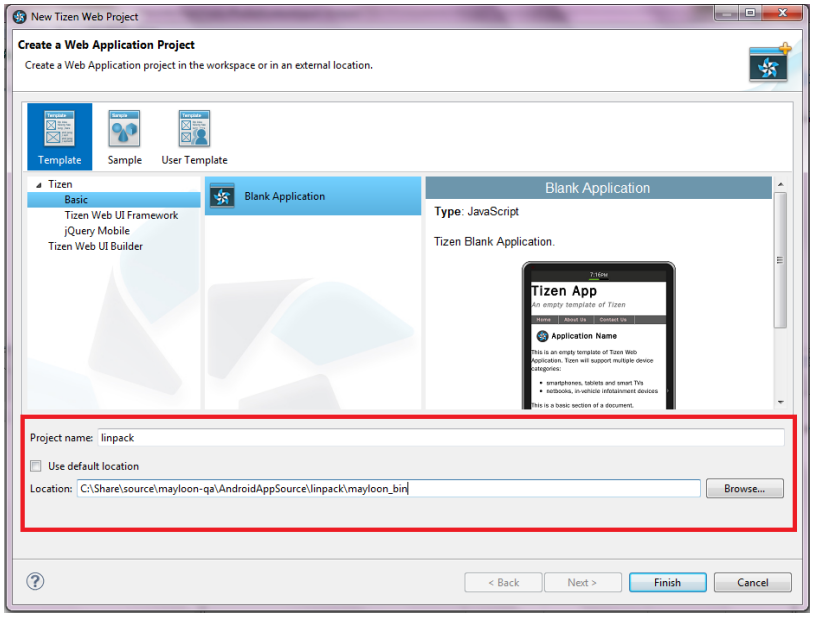
Pic23 Export application

5. Select the project you want to export and click Finish. By default the Tizen project will be exported to mayLoon_bin folder under your project folder.
6. Follow the page http://developer.tizen.org/downloads/sdk/installing-tizen-sdk to setup your Tizen development environment.
7. Open Tizen IDE and create a new Tizen web project



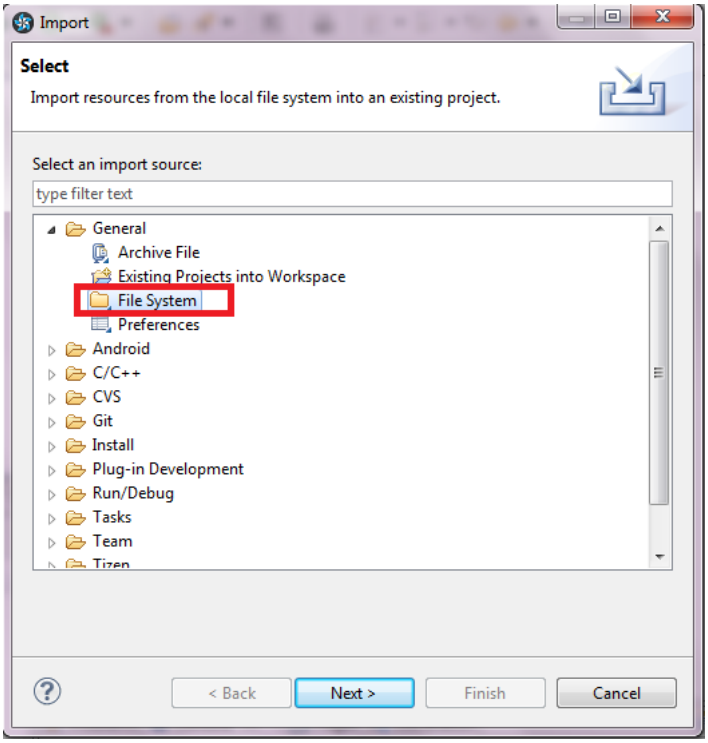
Pic24 Create Tizen web project

8. If you are working with Tizen IDE **2.0**, do not use default location, and set Location with content exported in step 5.

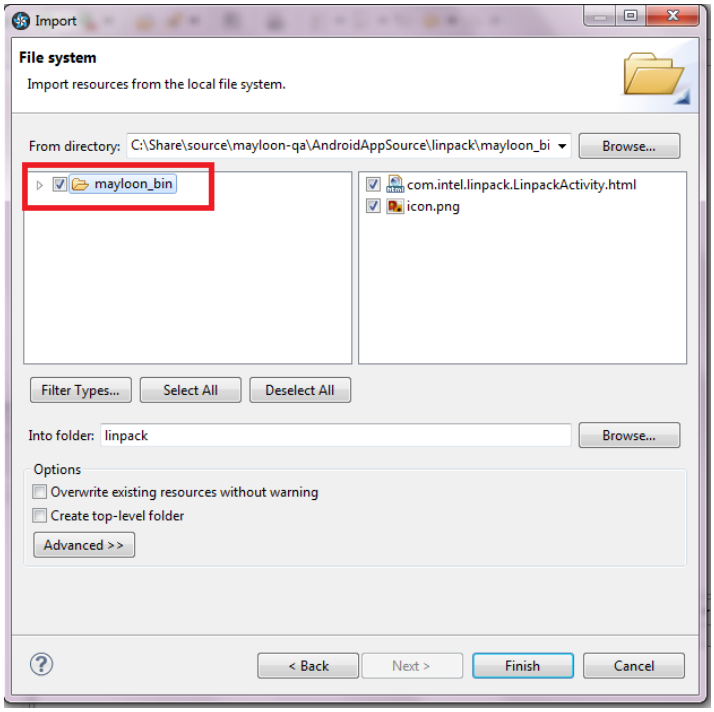


Pic25 Set project location

Otherwise if you are using Tizen IDE **2.1**, please create a new Tizen web project and import your exported MayLoon application to this newly created project as following shows. Please confirm if prompted to overwrite existing resources.

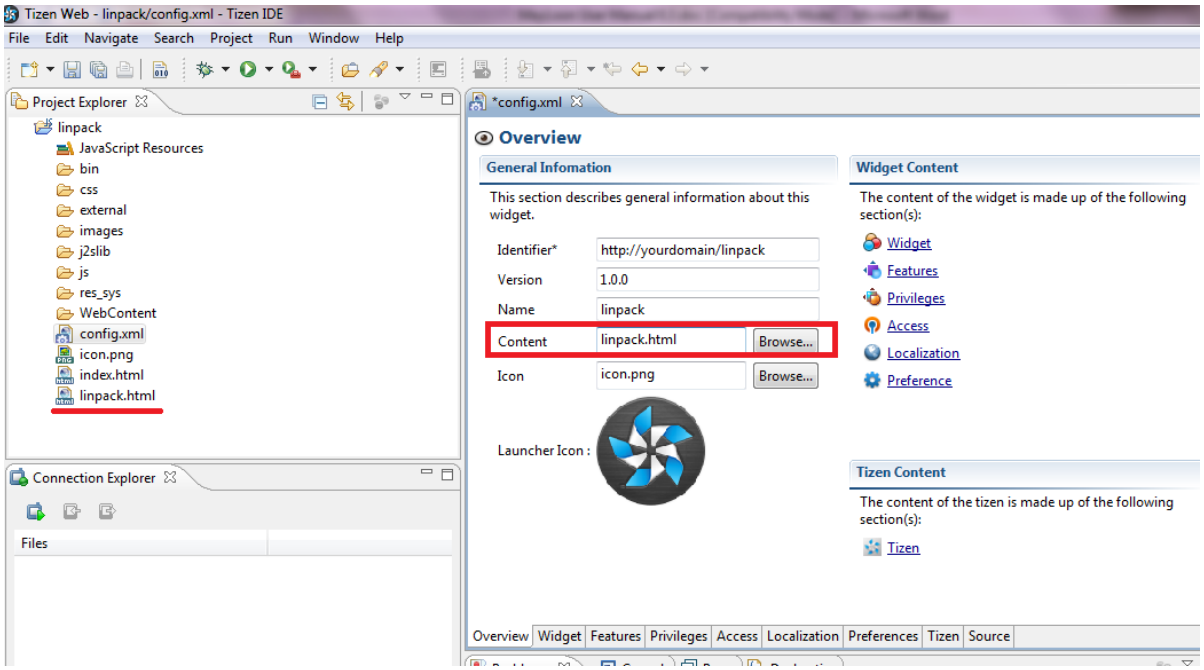


Pic26 Select import source



Pic27 Import from File System

9. Open config.xml in created project; modify index.html in Content tab to the generated html file.



Pic28 Set Content html file

10. Right click the project and click Run As->Tizen Web Simulator Application to run the project in Tizen web simulator.

11. If you want to run the application on a Tizen device, you should build the imported project in Tizen IDE, and a Tizen application widget (.wgt file) will be generated.

    Note: In order to speed up project build, please go to project Properties->Builders, then unselect Validator and JSLint Builder items. If you keep these builder options selected, the build process will be much longer. There might be compile errors in the project although they will not impact the generated package.

6 Supported APIs

Please refer to one of the MayloonAPISupport HTML documents to find the list of APIs supported by MayLoon alpha.

7 Unsupported Features

This chapter lists some of the features available from Android application framework while currently not available from MayLoon.

If some features are not supported, the corresponding API will not function as expected, but this does not necessarily mean that MayLoon will not be able to convert those Android applications.

- **Google service**

MayLoon has no access to Google service and APIs for now

- **TTS**

Text to speech APIs are not supported for now

- **Notification**

MayLoon currently doesn't have notification support.

- **Security model**

MayLoon does not support Android’s security model. Instead, the resulting HTML5 applications work in web browsers and web runtimes and are governed by their respective security models. Therefore behavior with security impact in MayLoon could be different from Android.

- **External storage**

MayLoon cannot access external storage on the devices yet.

- **Power management**

MayLoon does not support Android power management API and schemes.

- **GPS location provider**

MayLoon currently doesn't support GPS location provider.

- **Camera**

MayLoon applications cannot use camera yet.

- **Wallpaper**

The resulting HTML5 applications cannot operate on wallpaper.

- **AppWidget**
- **Predefined intents**

Intents is mechanism that Android implements to support message delivery between Android application and system. MayLoon currently does not support any of the predefined Intents – for example placing phone call or launching web browser.

- **Bluetooth operation**
- **Networking features**
- **Telephony features**
- **Vibrator**
- **File IO**
- **Sensors**

8 Known Incompatibilities and Limitations

8.1 Eclipse plugin

On Windows PC, if the Eclipse installation and the application are on different disk drives (C: /, D :/), the resulting HTML5 application might not be able to run.

8.2 Incompatibility between Java and JavaScript

This section lists the known incompatibilities when converting with MayLoon.

Currently, the MayLoon porting tool does not prompt the developer that any of the incompatibilities is identified in the original Android application.

As MayLoon translates between 2 different programming languages, some differences are inevitable:

- **NullPointerException**: In some APIs of the Android framework, a NullPointerException will be thrown when a null pointer is passed in for the API invocation. This mechanism is built into Java runtime. MayLoon applications do not always throw NullPointerException when it is required by the Android API specification.
- **Function overload:** Function overloading is a usual feature in Object Oriented languages. MayLoon implements majority of function overloading capability. However, consider the following Java class:

```
public class MyClass {

    public static void method(int intParam){
        System.out.println("int");
    }

    public static void method(float floatParam){
        System.out.println("float");
    }

    public static void method(String strParam){
        System.out.println("String");
    }

    public static void main(String[] args) {
        method(0);          // Call method(int)
        method(1.2f);       // Call method(float)
        method("Hello");    // Call method(String)
    }

}
```

One of the differences between JavaScript and Java is the numbering system. In JavaScript there is only one 'Number' type for all the numbers. Therefore, when converted to JavaScript, the application is not able to distinguish between

```
method(int intParam)
```

and

```
method(float floatParam).
```

When method(0) is invoked in the example, the program outputs "float" instead of "int". The application is able to distinguish between

```
method(String strParam)
```

from the other two though.

- **64-bit Bitwise Operators**: Bitwise operators are also translated from Java to JavaScript in a straightforward way. For example, bitwise AND

  a & b

  is translated into the same operation. In JavaScript, all numbers are stored as 64-bit (8-bytes) floating point numbers. However, JavaScript bitwise operators do not work on all 64 bits of the operands. Actually they only work on half of the bits - 32 bits. Therefore when the Android application performs bitwise operations on **long** integers, the behavior of the resulting HTML5 application might be different.

- **Multi-process and multi-thread**: Due to the single-threaded nature of JavaScript, MayLoon applications do not support multi-process or multi-thread. Threading behavior of MayLoon applications is undefined.

9 Upcoming Features

The following are some features we will add in upcoming releases:

- Basic HTTP networking
- SQLite database & Content Provider
- Simple services